Tree ensemble kernels for Bayesian optimization with known constraints over mixed-feature spaces

Anonymous Author(s)
Affiliation
Address
email

Abstract

Tree ensembles can be well-suited for black-box optimization tasks such as algorithm tuning and neural architecture search, as they achieve good predictive performance with little or no manual tuning, naturally handle discrete feature spaces, and are relatively insensitive to outliers in the training data. Two well-known challenges in using tree ensembles for black-box optimization are (i) effectively quantifying model uncertainty for exploration and (ii) optimizing over the piece-wise constant acquisition function. To address both points simultaneously, we propose using the kernel interpretation of tree ensembles as a Gaussian Process prior to obtain model variance estimates, and we develop a compatible optimization formulation for the acquisition function. The latter further allows us to seamlessly integrate known constraints to improve sampling efficiency by considering domain-knowledge in engineering settings and modeling search space symmetries, e.g. hierarchical relationships in neural architecture search. Our framework performs as well as state-of-the-art methods for unconstrained black-box optimization over continuous/discrete features and outperforms competing methods for problems combining mixed-variable feature spaces and known input constraints.

1 Introduction

Many black-box optimization problems contain feature relationships known *a priori* based on domain knowledge, such as hierarchies or constraints [34]. For example, hierarchical structures arise in neural architecture search [14, 47], where hyperparameters such as kernel size are only relevant if a convolutional layer is selected. Explicit constraints can also arise, e.g. matching kernel size and stride to input channel size and padding. In many cases, Bayesian optimization can incorporate known hierarchies and/or constraints, given that a suitable surrogate model is selected. To this end, Fromont et al. [16] and Nijssen and Fromont [35] impose a variable hierarchy by constraining the splitting order of decision trees, i.e. certain attributes must be selected before others.

Tree-based models, such as random forests or gradient-boosted trees, remain popular in many applications, as they inherit the innate ability of simple decision trees to seamlessly handle categorical and discrete input spaces. Moreover, they are highly parallelizable and scalable to high-dimensional data. Despite these modelling advantages, the deployment of tree-based models in Bayesian optimization has been limited by challenges pertaining to (i) quantifying prediction uncertainty and (ii) optimizing acquisition functions defined by their discontinuous response surfaces [41]. Early works, e.g. the popular SMAC algorithm [25], addressed (i) using empirical variance within a tree ensemble and (ii) via local and/or random search methods. Moreover, recent works [32, 33, 43] propose mixed-integer formulations for tree ensembles, enabling optimization over their mean functions.

Contributions. We present a mixed-integer second-order cone optimization formulation for tree kernel Gaussian processes in Section 3 and show that that the *tree agreement ratio*, i.e. the hyper-

Submitted to 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.

parameter introduced by the tree ensemble kernel, sufficiently represents the model uncertainty in Section 5.1. Section 5.2 shows that solving the mixed-integer second-order cone optimization problem considerably outperforms sampling-based strategies in Bayesian optimization. Our approach of using tree ensemble kernels as a Gaussian process prior is particularly useful for applications combining mixed-variable spaces and known input constraints.

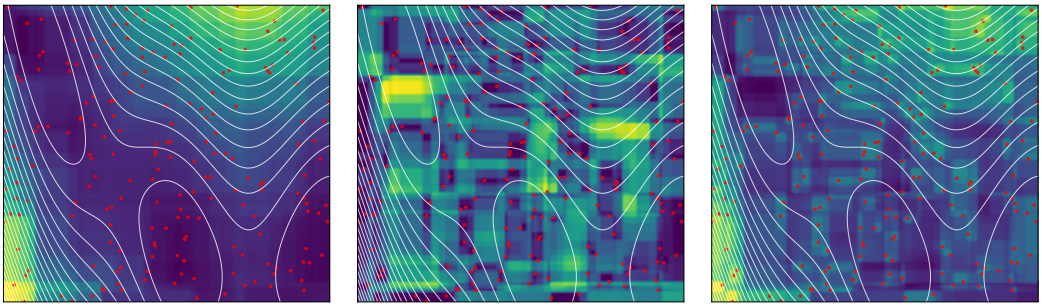
2 Related work

Bayesian optimization (BO) solves [15, 18, 41]: $\mathbf{x}_f^* \in \arg\min_{\mathbf{x}} f(\mathbf{x})$, where f is an expensive-to-evaluate black-box function that can be queried at inputs $\mathbf{x} \in \mathbb{R}$ to derive the optimal solution $\mathbf{x}_f^*$. BO iteratively updates a surrogate model of f and optimizes a corresponding acquisition function that balances exploitation and exploration. Minimizing the acquisition function produces a new query $\mathbf{x}^*$ which is evaluated and added to the set of observations. Gaussian processes (GPs) [38] are a common choice of BO surrogate due to their flexibility, e.g. domain-specific knowledge can be built into the GP prior via mean and kernel functions, and reliable uncertainty quantification to identify unexplored search areas. Open-source tools like BoTorch [1] implement BO with GP surrogates and offer a wide selection of kernels mainly suited for continuous search spaces. Prior work developed GPs with modified kernels to integrate discrete features [6, 13, 17, 21, 39] or considered conditional feature spaces [20, 26, 30, 31]. Besides GPs, tree ensemble-based surrogates show excellent performance for black-box optimization with mixed-variable settings, i.e. with continuous, integer and categorical variables, and for structured search spaces, e.g. hierarchical and conditional feature spaces [25].

Black-box optimization tools using tree ensembles, e.g. `SMAC` [25] and Scikit-Optimize (`SKOPT`) [22], are useful for applications such as neural architecture search (NAS) and algorithm tuning. Shahriari et al. [41] mention challenges in deploying tree ensembles for BO: (i) quantifying uncertainty for exploration purposes, and (ii) optimizing over the non-differentiable discrete acquisition function to determine the next query point. `SMAC` identifies uncertain search space regions using empirical variables across tree predictions of the random forest and optimizes the acquisition function combining local and random search. Bergstra et al. [3] proposes the Tree Parzen Estimator (TPE) to handle categorical variables and conditional structures by modeling individual input dimensions by a kernel density estimator. However, the TPE approach ignores dependencies between dimensions. For gradient-boosted tree ensembles, `SKOPT` derives uncertainty with quantile regression to fit two models for the 16th and 84th percentile and averages the predictions to estimate the standard deviation. For random forests, `SKOPT` uses an uncertainty strategy similar to `SMAC`. In general, `SKOPT` relies on random sampling to optimize the acquisition function.

Mišić [32] proposed a mixed-integer optimization formulation for tree ensemble mean functions that has been used in several applications [7, 33, 43, 44]. Besides improving the solution to an acquisition function, mixed-integer formulations also allow explicit consideration of input constraints to incorporate domain-knowledge. Some software tools, e.g. BoTorch, support linear equality and inequality constraints of continuous variables at the acquisition function optimization step while tree ensemble-based algorithms do not support input constraints. Papalexopoulos et al. [36] use ReLU neural networks as surrogate models and deploy a mixed-integer linear formulation to optimize the acquisition function. The approach relies on random initialization and stochasticity in the model training to allow for exploration. Daxberger et al. [11] handles mixed-variable search spaces by using a Bayesian linear regressor that uses an integer solver to search the discrete subspace. The authors introduce features capturing the discrete parts of the search space by using a BOCS model [2, 12] while continuous parts are handled with random Fourier features [37]. Genetic Algorithms (`GA`) are another class of algorithms, which deploy evolution-based selection heuristics to minimize black-box functions [27]. While there is no feasibility guarantee for input constraints, `GA` implementations like pymoo [4] support constraint optimization by minimizing constraint violation.

We compare our BO approach, which uses the kernel interpretation of tree ensembles as a Gaussian process prior, to: `SMAC`, the random forest and gradient-boosted tree versions of `SKOPT`, i.e. `SKOPT-RF`, and `SKOPT-GBRT`, in the Section 5 numerical studies as a baseline for other tree ensemble-based algorithms. We also compare against the default upper-confidence bound BO implementation of BoTorch, i.e. `UCB-MATERN`, and the default `GA` algorithm of pymoo to include black-box algorithms that partially support constraint optimization.



(a) $M(\mathbf{x})$ (b) $V(\mathbf{x})$ (c) $LCB(\mathbf{x})$

Figure 1: Tree kernel GP trained on 200 random points of the Branin function for the intervals $\mathbf{x} \in ([-5.0, 10.0], [0.0, 15.0])^\intercal$. Function values increase with the colour brightness and white contour lines indicate the true functional shape of the Branin function.

3 Technical background on prior work

3.1 Tree ensemble kernel as a Gaussian process prior

Exploring the search space requires quantifying the uncertainty of the underlying surrogate model. We use the kernel interpretation of tree ensembles based on random partitions [10, 48]. The tree kernel captures correlation between two input data points $(\mathbf{x}, \mathbf{x}') \in \mathbb{R}^n$:

$$k_{\text{Tree}}(\mathbf{x}, \mathbf{x}') = \sigma_0^2 \, |\mathcal{T}|^{-1} \, \mathbf{z}(\mathbf{x})^\intercal \mathbf{z}(\mathbf{x}') \tag{1}$$

To derive the tree kernel, we first train a gradient boosted tree ensemble $\mathcal{T}$ on data set $\mathbf{X} \in \mathbb{R}^{m \times n}$ with n denoting the dimensionality of the search space and m the size of the data set. Every tree t in the tree ensemble $\mathcal{T}$ maps inputs $\mathbf{x} \in \mathbb{R}^n$ onto a leaf l by sequentially evaluating splitting conditions. Each leaf l defines a subspace $\mathbf{x}_l \in \mathbb{R}^n$ restricted by active splits $s \in \mathbf{splits}(t)$. Two inputs $(\mathbf{x}, \mathbf{x}')$ are fully correlated in tree t if both end up in the leaf subspace $\mathbf{x}_{t,l}$ and uncorrelated otherwise. The Eq. (1) vector $\mathbf{z}(\mathbf{x})$ consists of binary elements $z_{t,l}$ indicating if leaf $l \in \mathcal{L}_t$ is active for input $\mathbf{x}$, with $\mathcal{L}_t$ denoting the set of all leaves in tree t. The inner product $\mathbf{z}(\mathbf{x})^\intercal \mathbf{z}(\mathbf{x}')$, normalized by the total number of trees $|\mathcal{T}|$, gives the ratio of trees in the ensemble for which $\mathbf{x}$ and $\mathbf{x}'$ fall into the same leaf. We modify the kernel by adding a trainable signal variance σ_0^2 [29]. Davies and Ghahramani [10] prove that the tree kernel is a suitable GP prior. The resulting (non-stationary and supervised) tree kernel describes a prior over piece-wise constant functions when used in a GP.

3.2 Posterior distribution

We approximate f as a Gaussian process with zero mean and kernel k_{Tree}: $f(\cdot) \sim \mathcal{GP}(0, k_{\text{Tree}})$. Since k_{Tree} is a valid Mercer kernel, the mean $M(\mathbf{x})$ and variance $V(\mathbf{x})$ of the GP at $x \in \mathbb{R}^n$ is [38]:

$$M(\mathbf{x}) = K_{\mathbf{x},\mathbf{X}} \, (K_{\mathbf{X},\mathbf{X}})^{-1} \, \mathbf{y} \tag{2a}$$

$$V(\mathbf{x}) = K_{\mathbf{x},\mathbf{x}} - K_{\mathbf{x},\mathbf{X}} \, (K_{\mathbf{X},\mathbf{X}})^{-1} \, K_{\mathbf{x},\mathbf{X}}^\intercal \tag{2b}$$

The Gram matrix $K_{\mathbf{X},\mathbf{X}} \in \mathbb{R}^{m \times m}$ has entries describing pairwise correlations computed based on the kernel function in Eq. (1) . The entries of vector $K_{\mathbf{x},\mathbf{X}} \in \mathbb{R}^m$ contain correlations between the input $\mathbf{x}$ and sampled data points, defined as $[k_{\text{Tree}}(\mathbf{x}, \mathbf{x}_1), k_{\text{Tree}}(\mathbf{x}, \mathbf{x}_2), \ldots, k_{\text{Tree}}(\mathbf{x}, \mathbf{x}_m)]$ with vectors $\mathbf{x}_i$ referring to rows in data set $\mathbf{X}$. Target values $\mathbf{y} \in \mathbb{R}^m$ are the corresponding observations for $\mathbf{X}$. Eq. (2) describes the noise-free case of the GP mean and variance. To fit a GP based on the tree kernel function, we usually require a noise term, i.e. a diagonal matrix $\sigma_y^2 I$ that is added to $K_{\mathbf{X},\mathbf{X}}$. We set hyperparameters σ_y^2 and σ_0^2 by maximizing the log marginal likelihood. Fig. 1 visualizes the Eq. (2) $M(\mathbf{x})$, $V(\mathbf{x})$ and the lower-confidence bound (LCB) [9] response surface. Fig. 1(a) shows that the tree kernel-based GP mean gives a good piecewise-constant approximation of the Branin function. Fig. 1(b) shows that variance peaks reveal areas where data is sparse, reliably identifying uncertainty in the underlying surrogate model. Finally, Fig. 1(c) shows how an acquisition function such as the LCB can effectively manage the exploitation-exploration trade-off.

3.3 Global optimization of tree ensembles

Mišić [32] proposes a mixed-integer linear optimization formulation that ensures that binary variables $z_{t,l}$ follow the logic of the tree ensemble:

$$\sum_{l \in \mathcal{L}_t} z_{t,l} = 1, \qquad \forall t \in \mathcal{T}, \quad (3a)$$

$$\sum_{l \in \mathbf{left}(s)} z_{t,l} \leq \sum_{j \in \mathbf{C}(s)} \nu_{\mathrm{V}(s),j}, \qquad \forall t \in \mathcal{T}, \forall s \in \mathbf{splits}(t), \quad (3b)$$

$$\sum_{l \in \mathbf{right}(s)} z_{t,l} \leq 1 - \sum_{j \in \mathbf{C}(s)} \nu_{\mathrm{V}(s),j}, \qquad \forall t \in \mathcal{T}, \forall s \in \mathbf{splits}(t), \quad (3c)$$

$$\sum_{j=1}^{K_i} \nu_{i,j} = 1, \qquad \forall i \in \mathcal{C}, \quad (3d)$$

$$\nu_{i,j} \leq \nu_{i,j+1}, \qquad \forall i \in \mathcal{N}, \forall j \in [K_i - 1], \quad (3e)$$

$$\nu_{i,j} \in \{0, 1\}, \qquad \forall i \in [n], \forall j \in [K_i], \quad (3f)$$

$$z_{t,l} \geq 0, \qquad \forall t \in \mathcal{T}, \forall l \in \mathcal{L}_t. \quad (3g)$$

Eq. (3a) guarantees exactly one active leaf l in leaf set $\mathcal{L}_t$ for tree t. Eqs. (3b)–(3c) ensure that binary variables $z_{t,l}$ are only active if all previous splits $\nu_{\mathrm{V}(s),j}$ are active. At any node in a given tree, $\mathbf{left}(s)$ and $\mathbf{right}(s)$ contain all leaves following the left and right branches, respectively. The mapping $\mathrm{V}(s)$ gives the splitting feature at node $s \in \mathbf{splits}(t)$ in tree t. Tree ensembles can handle continuous, integer and categorical data. Continuous splits s are defined by $x_{\mathrm{V}(s)} \leq v_{\mathrm{V}(s),j}$ conditions, where $v_{\mathrm{V}(s),j}$ is the learned splitting threshold. Therefore, $\mathbf{C}(s)$ only contains a single index j representing the threshold $v_{\mathrm{V}(s),j}$. Categorical splits s are characterized by (subsets of) categories available for feature $\mathrm{V}(s)$ and define a splitting condition based on the inclusion of $x_{\mathrm{V}(s)}$ in the category subset at split s. For categorical splits, $\mathbf{C}(s)$ includes the category indices comprising the category subset at split s. Eq. (3d) ensures that only one category is active per categorical variable $i \in \mathcal{C}$. Continuous splitting thresholds of all trees in the ensemble are ordered according to $v_{i,1} < v_{i,2} < ... < v_{i,K_i}$ with K_i denoting the index for the last split of continuous feature $i \in \mathcal{N}$. To enforce this order, Eq. (3e) ensures that binary variables $\nu_{i,j}$, corresponding to the split thresholds $v_{i,j}$, are activated sequentially. The model comprising Eqs. (3) has no direct dependency on $\mathbf{x}$ and is fully defined by binary variables indicating which splits and leaves of the tree model active. However, to allow the user to include extra equality and inequality constraints on the input vector $\mathbf{x}$, we bound the continuous variables based on the active splits by adding linking constraints [33]:

$$x_i \geq v_i^L + \sum_{j=1}^{K_i} (v_{i,j} - v_{i,j-1})(1 - \nu_{i,j}), \qquad \forall i \in \mathcal{N}, \quad (4a)$$

$$x_i \leq v_i^U + \sum_{j=1}^{K_i} (v_{i,j} - v_{i,j+1}) \nu_{i,j}, \qquad \forall i \in \mathcal{N}, \quad (4b)$$

$$x_i \in \left[v_i^L, v_i^U\right], \qquad \forall i \in \mathcal{N}, \quad (4c)$$

$$x_i = \{j \in [K_i] \mid \nu_{i,j} = 1\}, \qquad \forall i \in \mathcal{C}, \quad (4d)$$

For categorical variables, Eq. (4d) maps the indices of active categories onto x_i for $i \in \mathcal{C}$. These optimization formulations are implemented in open-source software ENTMOOT [43] and OMLT [7].

4 Tree ensemble kernels for Bayesian optimization

The technical details in Section 3 are insufficient to use the tree ensemble kernel in a Bayesian optimization framework. While Eqs. 3 and 4 allow optimization over the mean of an associated acquisition function [32], Bayesian optimization also requires quantifying model uncertainty for exploration. This section proposes a mixed-integer second-order cone optimization formulation to capture the standard deviation of a GP with a tree kernel prior. Combining this optimization

formulation with the already-developed mixed-integer formulation of the mean function, we derive the lower confidence bound (LCB) of the tree kernel-based GP. The advantage of deriving a mixed-integer second-order cone optimization formulation of the LCB acquisition function is that we can globally optimize the acquisition function. Additionally, we can easily incorporate explicit input constraints that capture domain knowledge and/or known search space relationships.

Our optimization problem, which includes the LCB acquisition function, is:

$$\mathbf{x}^*_{\text{lb}}, \mathbf{x}^*_{\text{ub}}, \mathbf{x}^*_{\text{cat}} \in \underset{\mathbf{x},\mathbf{z},\nu}{\arg\min}\ \mu(\mathbf{x}) - \kappa\sigma(\mathbf{x}), \tag{5a}$$

$$h(\mathbf{x}) = 0, \tag{5b}$$

$$g(\mathbf{x}) \leq 0, \tag{5c}$$

with $\mu(\mathbf{x})$ and $\sigma(\mathbf{x})$ denoting the surrogate model's mean prediction and standard deviation, respectively. Functions $h(\mathbf{x})$ and $g(\mathbf{x})$ are known constraints and handled similarly to [5, 45]. In our implementation, the constraints can be linear, quadratic, or polynomial. Hyperparameter κ controls the exploitation-exploration trade-off to determine the next black-box function query area. The solution to Eq. 5a is defined by $\mathbf{x}^*_{\text{lb}}$ and $\mathbf{x}^*_{\text{ub}}$ for non-categorical variables $i \in \mathcal{N}$, i.e. continuous and integer features, and $\mathbf{x}^*_{\text{cat}}$, a set of valid category subsets for categorical variables $i \in \mathcal{C}$. Two vectors ($\mathbf{x}^*_{\text{lb}}$ and $\mathbf{x}^*_{\text{ub}}$) define the non-categorical variables because the trees are piecewise-constant over intervals and the vectors define the lower and upper bounds of these intervals.

Next, we formalize the mean and variance of the tree kernel-based GP. Eqs. 6 and 2 are equivalent.

$$\mu(\mathbf{x}) = M(\mathbf{x}) = K_{\mathbf{x},\mathbf{X}}\ (K_{\mathbf{X},\mathbf{X}})^{-1}\ \mathbf{y} \tag{6a}$$

$$\sigma^2 = V(\mathbf{x}) = K_{\mathbf{x},\mathbf{x}} - K_{\mathbf{x},\mathbf{X}}\ (K_{\mathbf{X},\mathbf{X}})^{-1}\ K^{\intercal}_{\mathbf{x},\mathbf{X}} \tag{6b}$$

The Gram matrix $K_{\mathbf{X},\mathbf{X}}$ and target vector $\mathbf{y}$ are constants in the optimization model as these quantities only depend on the data set $\mathbf{X}$. The value $K_{\mathbf{x},\mathbf{x}}$ is directly related to signal variance hyperparameter σ_0 since there is full leaf overlap for two identical inputs:

$$K_{\mathbf{x},\mathbf{x}} = \sigma_0^2 \tag{7}$$

The vector $K_{\mathbf{x},\mathbf{X}}$ describes the kernel output of $\mathbf{x}$ with individual data points $\mathbf{x}_i$. We compute constant matrix $A \in \mathbb{R}^{m\times|\mathcal{L}|}$ after the gradient-boosted tree is trained but before solving the acquisition function. The entries of A are equal to 1 for all active leaves l of data point i and 0 otherwise. Given matrix A, Eqs. (8) capture the kernel output by summing up all binary variables $z_{t,l}$ that are active for data point i.

$$K_{\mathbf{x},\mathbf{X}} = [k_{\text{Tree}}(\mathbf{x},\mathbf{x}_1), k_{\text{Tree}}(\mathbf{x},\mathbf{x}_2), \ldots, k_{\text{Tree}}(\mathbf{x},\mathbf{x}_m)] \tag{8a}$$

$$k_{\text{Tree}}(\mathbf{x},\mathbf{x}_i) = \sigma_0^2\ |\mathcal{T}|^{-1} \sum_{t\in\mathcal{T}} \sum_{l\in\mathcal{L}_t} A_{i,l} z_{t,l} \qquad \forall i \in [m] \tag{8b}$$

To get an intuition for Eqs. (8), note that the largest possible value for each $k_{\text{Tree}}(\mathbf{x},\mathbf{x}_i)$ is σ_0^2 (if the corresponding entries of matrix A are all equal to 1 and there is full leaf overlap) and the smallest possible value for each $k_{\text{Tree}}(\mathbf{x},\mathbf{x}_i)$ is 0 (if there is no leaf overlap). In general, values of each element of vector $K_{\mathbf{x},\mathbf{X}}$ will range between 0 and σ_0^2: higher values in the elements of $K_{\mathbf{x},\mathbf{X}}$ indicate a higher degree of overlap between the next query location $\mathbf{x}$ and data point $\mathbf{x}_i$ in the set $\mathbf{X}$.

Without considering additional tree model constraints h and g, the resulting acquisition function is a mixed-integer quadratic optimization problem. The optimization problem is *mixed-integer* because of binary variables $\boldsymbol{\nu}$ and *quadratic* because of Eq. (6b). The quadratic Eq. (6b) components are σ^2 and the terms $k^2_{\text{Tree}}(\mathbf{x},\mathbf{x}_i)$ arising from the inner product of $K_{\mathbf{x},\mathbf{X}}$ with itself. We only require one direction of the Eq. (6b) equality ($\leq$) and re-write Eq. (6b) as a second-order cone constraint. Solver Gurobi 9 [19] automatically finds that Eq. (6b) (with $\leq$ rather than $=$) can be represented as a second-order cone and makes the appropriate algorithm modifications, e.g. as described by [24, 46].

The acquisition function, with Objective 5a and Constraints (3)–(8b), is a mixed-integer second-order cone program which can be solved with optimization solvers. A valid solution to the proposed model is a set of active leaves $\mathcal{L}^*$ and the intersection of all leaf subspaces $[\mathbf{x}^*_{\text{lb}}, \mathbf{x}^*_{\text{ub}}], \mathbf{x}^*_{\text{cat}}$.

To derive the next black-box function query point $\mathbf{x}^*$, we propose some heuristics. The acquisition function value is constant for $\boldsymbol{x} \in [\mathbf{x}^*_{\text{lb}}, \mathbf{x}^*_{\text{ub}}], \mathbf{x}^*_{\text{cat}}$, i.e. all contained points are equivalent from the

perspective of the tree kernel-GP. For the continuous and integer features, we observe that tree models tend to learn split thresholds close to training data points. We propose the center of $[\mathbf{x}^*_{\text{lb}}, \mathbf{x}^*_{\text{ub}}]$ as the next query point $\mathbf{x}^*$ for continuous and integer features:

$$x^*_{\text{mid},i} = \frac{1}{2}\left(x^*_{lb,i} + x^*_{ub,i}\right), \forall i \in \mathcal{N}. \tag{9}$$

For integer features with a fractional mid-point between the upper and lower bound, we randomly select its floor or ceiling. For categorical features, we sample from the subset of available categories:

$$x^*_{\text{mid},i} = \text{uniform}(\mathbf{x}^*_{\text{cat}}), \forall i \in \mathcal{C}. \tag{10}$$

For the unconstrained case, we use $\mathbf{x}^*_{\text{mid}}$ as the next query point $\mathbf{x}^*$. When additional input constraints $h(\mathbf{x})$ and/or $g(\mathbf{x})$ are given, we know that at least one point in $[\mathbf{x}^*_{\text{lb}}, \mathbf{x}^*_{\text{ub}}], \mathbf{x}^*_{\text{cat}}$ is feasible even if the heuristic $\mathbf{x}^*_{\text{mid}}$ is infeasible. To repair the solution $\mathbf{x}^*_{\text{mid}}$, we project it onto the feasible space:

$$\mathbf{x}^* \in \underset{\mathbf{x}\in[\mathbf{x}^*_{\text{lb}},\mathbf{x}^*_{\text{ub}}],\mathbf{x}^*_{\text{cat}}}{\arg\min} \sum_{i\in\mathcal{N}} \left(x^*_{\text{mid},i} - x_i\right)^2 - \sum_{i\in\mathcal{C}} \sum_{j\in x^*_{\text{mid},i}} \nu_{i,j} \tag{11a}$$

$$h(\mathbf{x}) = 0, \tag{11b}$$

$$g(\mathbf{x}) \leq 0. \tag{11c}$$

Eq. (11) projects $\mathbf{x}^*_{\text{mid}}$ onto the feasible set defined by $h(\mathbf{x})$ and $g(\mathbf{x})$. The time complexity of solving Eq. 11, and that of solving Eqs. (3)–(6) for the search space $[\mathbf{x}^*_{\text{lb}}, \mathbf{x}^*_{\text{ub}}], \mathbf{x}^*_{\text{cat}}$, are both NP-hard. However, we found that Gurobi 9 often solves both problems to ϵ-global optimality for moderately-sized tree models in less time compared to random sampling.

Hyperparameters. We train the kernel hyperparameters signal variance σ_0^2 and noise σ_y^2 by maximizing the log marginal likelihood. Additional hyperparameters are introduced by the gradient-boosted tree ensemble, i.e. maximum tree depth and number of trees, and through κ in the LCB. For the Section 5 numerical studies, we leave κ and gradient-boosting hyperparameters constant and only increase maximum tree depth and number of trees for the high-dimensional CIFAR NAS benchmark to capture more complicated interactions. The appendix reports specific values for all hyperparameters.

Limitations. The method suffers from standard BO challenges where the tree kernel may not be a good prior for the underlying black-box function, e.g. purely continuous feature spaces. More limitations arise from solving an NP-hard problem to global optimality when working with large data sets in high-dimensional search spaces. For cases where solving the NP-hard problem is too difficult but BO is still applicable, Gurobi 9 can typically develop good feasible solutions as a heuristic.

5 Numerical studies

This section empirically evaluates the performance of tree kernel-based GPs using a wide variety of synthetic and real-world benchmark problems. We show (i) the ability of tree kernels to capture uncertainty of the underlying tree ensemble, (ii) the advantage of using global vs. local strategies for optimizing the acquisition function and (iii) the proposed algorithm's superior performance in cases with constrained search spaces and mixed variable types. `LEAF-GP` denotes the Section 4 proposed algorithm, and Section 2 outlines the baseline of methods we compare against. For every run, we visualize the median and confidence intervals of the first and third quartile based on 20 individual runs with varying random seeds. Further technical details can be found in the appendix.

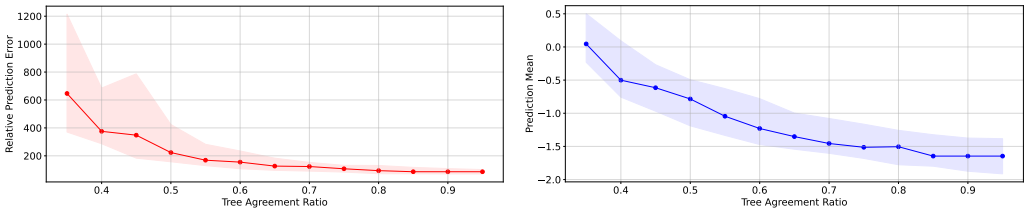
5.1 Uncertainty metric

The tree kernel-based GP uses the leaf overlap measure to quantify correlation between two inputs. In BO, such measures help identify unexplored areas where correlation to existing training data is low and we expect inaccurate model predictions. To empirically test the tree kernel's capability of identifying uncertainty in the underlying tree ensemble, we change the optimization formulation:

$$\mathbf{x}^*_{\text{lb}}, \mathbf{x}^*_{\text{ub}}, \mathbf{x}^*_{\text{cat}} \in \underset{\mathbf{x},\mathbf{z},\nu}{\arg\min}\ \mu(\mathbf{x}), \tag{12a}$$

$$\text{s.t. Eq. (3), Eq. (4), Eq. (6), Eq. (7), Eq. (8)} \tag{12b}$$

$$|\mathcal{T}|^{-1} \sum_{t\in\mathcal{T}} \sum_{l\in\mathcal{L}_t} A_{i,l} z_{t,l} \leq R,\ \forall i \in [m] \tag{12c}$$



(a) Relative Model Error (b) Prediction Mean

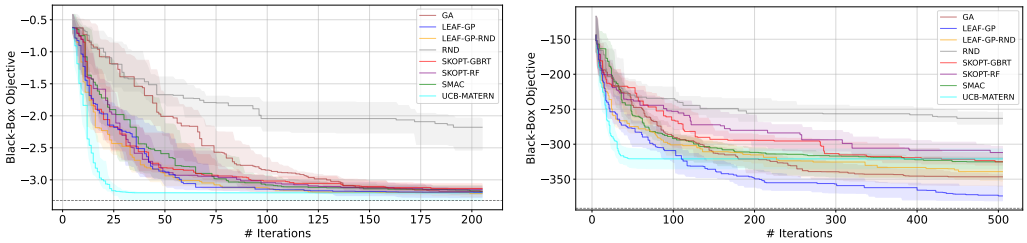
Figure 2: The relative prediction error (Eq. 13) and model prediction mean over the maximum tree agreement ratio R for benchmark problem Rastrigin (10D). Plot shows the median line and confidence intervals (first and third quartile) from 20 random seeds. Section 5.1 provides more details.

Eq. (12a) minimizes over the mean prediction of the tree kernel-based GP with Eq. (12c) restricting the ratio of tree agreement for the black-box query point $\mathbf{x}^*$ defined as the Eq. (9) center of the optimal area $\mathbf{x}^*_{\text{lb}}, \mathbf{x}^*_{\text{ub}}, \mathbf{x}^*_{\text{cat}}$. A solution to problem Eq. (12) guarantees a maximum leaf overlap of R between $\mathbf{x}^*$ and all available data points $\mathbf{X}$. Here, we evaluate values of $R \in [0.35, 1.0]$ with increments of 0.05 and compute the model error according to:

$$\epsilon_{\text{error}} = \left| \frac{\mu(\mathbf{x}^*) - f_{\text{true}}(\mathbf{x}^*)}{\mu(\mathbf{x}^*)} \right| \tag{13}$$

Fig. 2 shows results for the Rastrigin [42] benchmark. With smaller values of R, the Eq. (12) optimization problem becomes more restrictive, leading to larger objective values for solution $\mathbf{x}^*$. Fig. 2(a) shows that a growing leaf overlap, i.e. larger R, reduces the model error defined in Eq. (13). This suggests that the kernel works as intended and that low kernel correlation can reveal search space areas with high model uncertainty. The appendix provides results for additional benchmarks.

5.2 Local vs. global acquisition optimization



(a) Hartmann (6D) (b) Styblinski-Tang (10D)

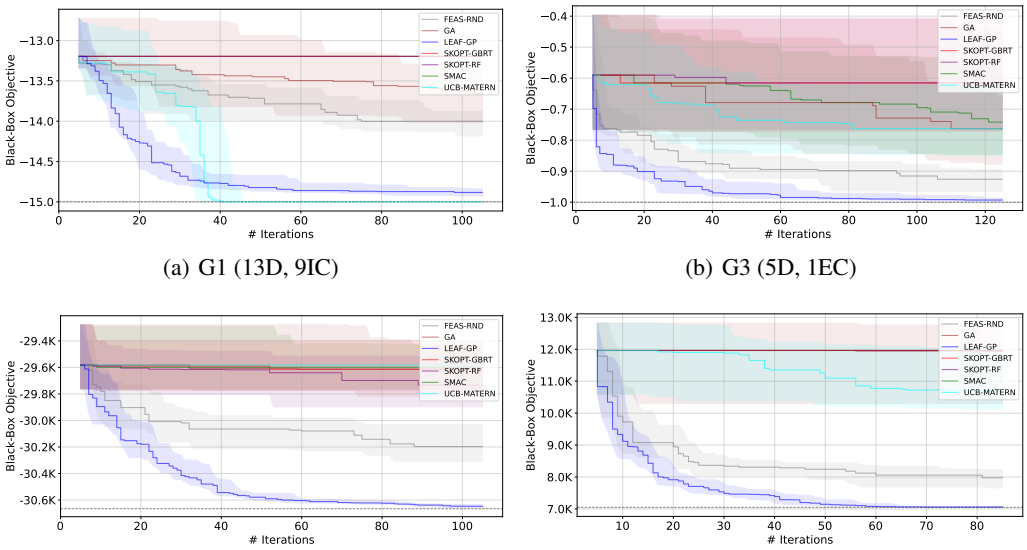
Figure 3: Black-box optimization progress of `LEAF-GP` vs. baseline. Plot shows the median line and confidence intervals (first and third quartile) from 20 random seeds. Section 5.2 provides more details

This section (i) compares `LEAF-GP` to other state-of-the-art algorithms for common benchmarks and (ii) shows the advantage of global vs. local strategies for optimizing the acquisition function. We introduce a variation of the proposed algorithm, `LEAF-GP-RND`, which optimizes the same acquisition function as `LEAF-GP`, but uses random sampling instead of Gurobi 9. Fig. 3 shows results for Hartmann (6D) [42] and Styblinski-Tang (10D) [42]. We do not expect tree model-based approaches to perform well since both benchmark functions are continuous. However, the benchmarks show that all approaches perform similarly for Hartmann (6D) with `UCB-MATERN` improving the black-box objective at the fastest rate. For Styblinski-Tang (10D), `LEAF-GP` significantly outperforms other algorithms. Although `LEAF-GP` is not specialized to this setting, observe that it performs roughly with the state-of-the-art on these continuous, unconstrained optimization problems.

Although sampling-based `LEAF-GP-RND` and optimization-based `LEAF-GP` perform similarly on the smaller Hartmann (6D), `LEAF-GP` is particularly strong on the higher-dimensional Styblinski-Tang (10D) benchmark function, and two additional benchmarks in the appendix. Moreover, `LEAF-GP-RND`

does not support explicit input constraints and performs particularly bad for BO with known constraints. For the following constrained benchmarks, we remove `LEAF-GP-RND` from the comparison. The appendix has additional details and results.

5.3 Constrained spaces



(a) G1 (13D, 9IC)

(b) G3 (5D, 1EC)

(c) G4 (10D, 6IC)

(d) Alkylation (7D, 14IC)

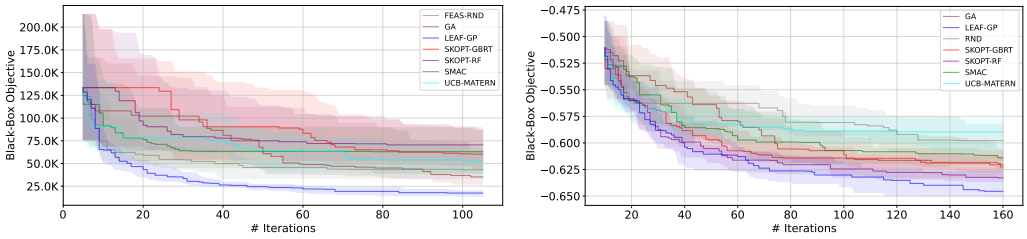
Figure 4: Feasible black-box optimization progress of `LEAF-GP` vs. baseline. Plot shows the median line and confidence intervals (first and third quartile) from 20 random seeds. Confidence intervals are neglected for methods that cannot improve the initial training data. Figure subtitles give the function name and number of: dimensions (D), equality constraints (EC), and inequality constraints (IC). Section 5.3 provides more details.

This section presents numerical benchmarks with known input constraints. The acquisition optimization strategy of `LEAF-GP` allows explicit consideration of input constraints, i.e. logical and convex/non-convex n-th degree polynomial equality and inequality constraints for mixed variable spaces. The `UCB-MATERN` implementation of BoTorch supports only linear equality and inequality constraints at the acquisition optimization step. The `GA` pymoo algorithm has an interface for callable constraint functions which are considered when generating new generations of candidate points. When a method does not support the specific input constraints, we penalize the objective:

$$f_{\text{penalty}} = \lambda \left(\max(g(\mathbf{x}), 0)^2 + h(\mathbf{x})^2 \right), \tag{14}$$

where $g(\mathbf{x}) \leq 0$ and $h(\mathbf{x}) = 0$ are inequality and equality constraints, respectively. This penalty strategy allows methods that do not support explicit constraints to still produce feasible points given enough iterations. Eq. (14) introduces the hyperparameter λ which weights the penalty, we test values $\lambda \in \{1, 10, 100\}$ and only plot the best run for all methods that rely on constraint penalization. To initialize every method with feasible points, we draw random samples from a uniform distribution within the bounds that define the search space and compute the closest feasible point similar to Eq. (11). For constraint benchmarks, we introduce `FEAS-RANDOM` which simply projects random samples onto the set of feasible points that satisfy $h(\mathbf{x})$ and $g(\mathbf{x})$. Fig. 4 plots feasible solutions to four different continuous benchmark problems, G1, G3, G4 [23], and Alkylation [40]. The G1 benchmark has linear inequality constraints only, which are supported by `UCB-MATERN` and `LEAF-GP`. Fig. 4(a) shows `LEAF-GP` making quick progress at the beginning with `UCB-MATERN` catching up towards the end. G3 and G4 have different combinations of nonlinear constraints, and `LEAF-GP` significantly outperforms competing methods and random feasible sampling. The Alkylation benchmark determines optimal operating conditions for a simplified alkylation process considering complicated nonlinear constraints representing economic, physical and performance limits. Again, `LEAF-GP` outperforms other methods, which often struggle to find feasible solutions. More details regarding the presented benchmark problems are given in the appendix.

5.4 Mixed-variable spaces



(a) Pressure Vessel (4D, 3IC) (b) CIFAR NAS (29D)

Figure 5: Feasible black-box optimization comparing `LEAF-GP` vs. baseline. Plot shows median line and confidence intervals (first and third quartile) from 20 random seeds. Figure subtitles give the number of dimensions (D) and inequality constraints (IC). Section 5.4 provides details.

We now consider spaces that exhibit mixtures of continuous, integer and categorical variables for constrained problems. Tree model-based algorithms naturally support categorical variables by replacing continuous splits with categorical splits. For all methods that do not support categorical features, we use one-hot encoding. The category with the highest corresponding auxiliary variable value is chosen for the subsequent query point. On the Pressure Vessel (4D) [8] benchmark, which comprises two continuous features, two integer features and three inequality constraints, Fig. 5(a) shows that `LEAF-GP` outperforms other black-box optimization methods.

The CIFAR NAS (29D) is a high-dimensional benchmark problem with one continuous, 23 integer and five categorical variables. It describes properties of different layers and training hyperparameters for a convolutional neural network (CNN) trained on CIFAR-10 [28]. The problem's search space is hierarchical, as different layers, i.e. convolutional and fully-connected layers, can be activated. Properties describing a layer are only relevant if the layer is active. For this problem, we introduce two types of constraints to guide the `LEAF-GP`: (i) constraints that allow for feasible network designs, i.e. different values for padding, stride, kernel size and max pooling in earlier layers affects what kernel size is feasible in following layers and (ii) constraints that ensure that hierarchies are respected. Towards the former, we include constraints to ensure the output size of every convolutional layer is positive. In general, tree models are particularly good at capturing hierarchical search space structures. To enforce hierarchical relationships during the acquisition optimization step, we introduce indicator constraints that force layer properties to take predefined default values if the layer is inactive. This ensures that the optimizer avoids leaves that infer splitting conditions from feature properties of inactive layers, thus effectively reducing the search space size. To expedite tests, we train the CNN on half of the CIFAR-10 training data and optimize for test accuracy. Fig. 5(b) summarizes the results of this study and shows that tree model-based algorithms generally outperform `UCB-MATERN`. Utilizing search space constraints, `LEAF-GP` finds the neural architecture with the highest test accuracy. Further details about this benchmark are given in the appendix.

6 Conclusion

We present a framework for black-box optimization based on tree kernel Gaussian processes that simultaneously allows (i) reliable uncertainty quantification of mixed feature spaces and (ii) incorporation of explicit input constraints. Although these two needs have been considered separately, we are able to address both simultaneously through the mixed-integer second-order cone formulation of the acquisition function. The numerical studies show that the proposed strategy performs competitively with state-of-the-art algorithms for unconstrained problems and significantly outperforms existing methods for constrained benchmarks, especially those with mixed feature spaces. We use optimization constraints together with the acquisition functions to incorporate domain knowledge and leverage hierarchical search space structures, e.g. for neural architecture search.

References

[1] Maximilian Balandat, Brian Karrer, Daniel Jiang, Samuel Daulton, Ben Letham, Andrew G Wilson, and Eytan Bakshy. Botorch: a framework for efficient Monte-Carlo Bayesian optimization. *Advances in neural information processing systems*, 33:21524–21538, 2020.

[2] Ricardo Baptista and Matthias Poloczek. Bayesian optimization of combinatorial structures. In *International Conference on Machine Learning*, pages 462–471. PMLR, 2018.

[3] James Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. Algorithms for hyper-parameter optimization. *Advances in neural information processing systems*, 24, 2011.

[4] J. Blank and K. Deb. pymoo: Multi-objective optimization in python. *IEEE Access*, 8:89497–89509, 2020.

[5] Fani Boukouvala and Marianthi G. Ierapetritou. Derivative-free optimization for expensive constrained problems using a novel expected improvement objective function. *AIChE Journal*, 60(7):2462–2474, 2014.

[6] Poompol Buathong, David Ginsbourger, and Tipaluck Krityakierne. Kernels over sets of finite sets using rkhs embeddings, with application to bayesian (combinatorial) optimization. In *International Conference on Artificial Intelligence and Statistics*, pages 2731–2741. PMLR, 2020.

[7] Francesco Ceccon, Jordan Jalving, Joshua Haddad, Alexander Thebelt, Calvin Tsay, Carl D Laird, and Ruth Misener. Omlt: Optimization & machine learning toolkit. *arXiv preprint arXiv:2202.02414*, 2022.

[8] Carlos A Coello Coello and Efrén Mezura Montes. Constraint-handling in genetic algorithms through the use of dominance-based tournament selection. *Advanced Engineering Informatics*, 16(3):193–203, 2002.

[9] Dennis D Cox and Susan John. A statistical method for global optimization. In *Proceedings 1992 IEEE International Conference on Systems, Man, and Cybernetics*, pages 1241–1246. IEEE, 1992.

[10] Alex Davies and Zoubin Ghahramani. The random forest kernel and other kernels for big data from random partitions. *arXiv preprint arXiv:1402.4293*, 2014.

[11] Erik Daxberger, Anastasia Makarova, Matteo Turchetta, and Andreas Krause. Mixed-variable bayesian optimization. *arXiv preprint arXiv:1907.01329*, 2019.

[12] Aryan Deshwal, Syrine Belakaria, and Janardhan Rao Doppa. Scalable combinatorial bayesian optimization with tractable statistical models. *arXiv preprint arXiv:2008.08177*, 2020.

[13] Aryan Deshwal, Syrine Belakaria, and Janardhan Rao Doppa. Bayesian optimization over hybrid spaces. In *International Conference on Machine Learning*, pages 2632–2643. PMLR, 2021.

[14] Thomas Elsken, Jan Hendrik Metzen, and Frank Hutter. Neural architecture search: A survey. *The Journal of Machine Learning Research*, 20(1):1997–2017, 2019.

[15] P. I. Frazier. A tutorial on Bayesian optimization. *ArXiv*, 1807.02811, 2018.

[16] Élisa Fromont, Hendrik Blockeel, and Jan Struyf. Integrating decision tree learning into inductive databases. In *International Workshop on Knowledge Discovery in Inductive Databases*, pages 81–96. Springer, 2006.

[17] Eduardo C Garrido-Merchán and Daniel Hernández-Lobato. Dealing with categorical and integer-valued variables in Bayesian optimization with Gaussian processes. *Neurocomputing*, 380:20–35, 2020.

[18] Stewart Greenhill, Santu Rana, Sunil Gupta, Pratibha Vellanki, and Svetha Venkatesh. Bayesian optimization for adaptive experimental design: A review. *IEEE access*, 8:13937–13948, 2020.

[19] Gurobi Optimization, LLC. Gurobi Optimizer Reference Manual, 2022. URL https://www.gurobi.com.

[20] Eric Han, Ishank Arora, and Jonathan Scarlett. High-dimensional bayesian optimization via tree-structured additive models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 7630–7638, 2021.

[21] Florian Häse, Matteo Aldeghi, Riley J Hickman, Loïc M Roch, and Alán Aspuru-Guzik. Gryffin: An algorithm for Bayesian optimization of categorical variables informed by expert knowledge. *Applied Physics Reviews*, 8(3):031406, 2021.

[22] Tim Head, MechCoder, Gilles Louppe, Iaroslav Shcherbatyi, fcharras, Zé Vinícius, cmmalone, Christopher Schröder, nel215, and Nuno Campos et al. scikit-optimize/scikit-optimize: v0.5.2. March 2018. doi: 10.5281/zenodo.1207017. URL https://doi.org/10.5281/zenodo.1207017.

[23] A. Hedar. Global Optimization Test Problems. Retrieved May 18, 2022, from http://www-optima.amp.i.kyoto-u.ac.jp/member/student/hedar/Hedar_files/TestGO.htm.

[24] Hassan Hijazi, Pierre Bonami, and Adam Ouorou. An outer-inner approximation for separable mixed-integer nonlinear programs. *INFORMS Journal on Computing*, 26(1):31–44, 2014.

[25] F. Hutter, H. H. Hoos, and K. Leyton-Brown. Sequential Model-Based Optimization for General Algorithm Configuration. In *Proceedings of the 5th International Conference on Learning and Intelligent Optimization*, pages 507–523. Springer-Verlag, 2011.

[26] Rodolphe Jenatton, Cedric Archambeau, Javier González, and Matthias Seeger. Bayesian optimization with tree-structured dependencies. In *International Conference on Machine Learning*, pages 1655–1664. PMLR, 2017.

[27] Y. Jin and J. Branke. Evolutionary optimization in uncertain environments-a survey. *IEEE Transactions on Evolutionary Computation*, 9(3):303–317, 2005.

[28] Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. Cifar-10 (canadian institute for advanced research). URL http://www.cs.toronto.edu/~kriz/cifar.html.

[29] Donghoon Lee, Hyunsin Park, and Chang D Yoo. Face alignment using cascade gaussian process regression trees. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4204–4212, 2015.

[30] Julien-Charles Lévesque, Audrey Durand, Christian Gagné, and Robert Sabourin. Bayesian optimization for conditional hyperparameter spaces. In *2017 International Joint Conference on Neural Networks (IJCNN)*, pages 286–293. IEEE, 2017.

[31] Xingchen Ma and Matthew Blaschko. Additive tree-structured covariance function for conditional parameter spaces in Bayesian optimization. In *International Conference on Artificial Intelligence and Statistics*, pages 1015–1025. PMLR, 2020.

[32] Velibor V Mišić. Optimization of tree ensembles. *Operations Research*, 68(5):1605–1624, 2020.

[33] Miten Mistry, Dimitrios Letsios, Gerhard Krennrich, Robert M Lee, and Ruth Misener. Mixed-integer convex nonlinear optimization with gradient-boosted trees embedded. *INFORMS Journal on Computing*, 33(3):1103–1119, 2021.

[34] Géraldin Nanfack, Paul Temple, and Benoît Frénay. Constraint enforcement on decision trees: a survey. *ACM Computing Surveys (CSUR)*, 2022.

[35] Siegfried Nijssen and Elisa Fromont. Mining optimal decision trees from itemset lattices. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 530–539, 2007.

[36] Theodore Papalexopoulos, Christian Tjandraatmadja, Ross Anderson, Juan Pablo Vielma, and David Belanger. Constrained discrete black-box optimization using mixed-integer programming. *arXiv preprint arXiv:2110.09569*, 2021.

[37] Ali Rahimi and Benjamin Recht. Random features for large-scale kernel machines. *Advances in neural information processing systems*, 20, 2007.

[38] C. E. Rasmussen and C. K. I. Williams. *Gaussian Processes for Machine Learning*. The MIT Press, 2006.

[39] Binxin Ru, Ahsan Alvi, Vu Nguyen, Michael A Osborne, and Stephen Roberts. Bayesian optimisation over multiple continuous and categorical inputs. In *International Conference on Machine Learning*, pages 8276–8285. PMLR, 2020.

[40] RN Sauer, AR Colville, and CW Burwick. Computer points way to more profits. *Hydrocarbon Processing*, 84(2), 1964.

[41] B. Shahriari, K. Swersky, Z. Wang, R. P. Adams, and N. de Freitas. Taking the Human Out of the Loop: A Review of Bayesian Optimization. *Proceedings of the IEEE*, 104(1):148–175, 2016.

[42] S. Surjanovic and D. Bingham. Virtual library of simulation experiments: Test functions and datasets. Retrieved May 18, 2022, from `http://www.sfu.ca/~ssurjano`.

[43] Alexander Thebelt, Jan Kronqvist, Miten Mistry, Robert M Lee, Nathan Sudermann-Merx, and Ruth Misener. Entmoot: a framework for optimization over ensemble tree models. *Computers & Chemical Engineering*, 151:107343, 2021.

[44] Alexander Thebelt, Calvin Tsay, Robert M Lee, Nathan Sudermann-Merx, David Walz, Tom Tranter, and Ruth Misener. Multi-objective constrained optimization for energy applications via tree ensembles. *Applied Energy*, 306:118061, 2022.

[45] Anh Tran, Jing Sun, John M. Furlan, Krishnan V. Pagalthivarthi, Robert J. Visintainer, and Yan Wang. pBO-2GP-3B: A batch parallel known/unknown constrained Bayesian optimization with feasibility classification and its applications in computational fluid dynamics. *Computer Methods in Applied Mechanics and Engineering*, 347:827–852, 2019.

[46] Juan Pablo Vielma, Iain Dunning, Joey Huchette, and Miles Lubin. Extended formulations in mixed integer conic quadratic programming. *Mathematical Programming Computation*, 9(3): 369–418, 2017.

[47] Chris Ying, Aaron Klein, Eric Christiansen, Esteban Real, Kevin Murphy, and Frank Hutter. Nas-bench-101: Towards reproducible neural architecture search. In *International Conference on Machine Learning*, pages 7105–7114. PMLR, 2019.

[48] Azar Zafari, Raul Zurita-Milla, and Emma Izquierdo-Verdiguier. Evaluating the performance of a random forest kernel for land cover classification. *Remote sensing*, 11(5):575, 2019.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes] See Abstract and Section 1.
   (b) Did you describe the limitations of your work? [Yes] See Section 4.
   (c) Did you discuss any potential negative societal impacts of your work? [No] There is no obvious societal impact of the method proposed.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] We submit the code to run all experiments for the proposed method.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] See end of Section 4 and supplemental material.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] We run 20 random seeds for every experiment and report median and confidence intervals, i.e. first and third quartile. See Section 5.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] We report the computational resources used to conduct the experiments in the supplementary material.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes] Yes, we cite all libraries that were used when preparing the numerical studies. For more information, we refer to the supplemental material.
   (b) Did you mention the license of the assets? [No] License information for all assets can be found in the documentation of each of the assets.
   (c) Did you include any new assets either in the supplemental material or as a URL? [No] The code to reproduce the paper's main results will be published open-source after the peer review process.
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]